Learning Markov models using ADDs

Workshop in honour of Anna Ingólfsdóttir

Giovanni Bacci

joint work with

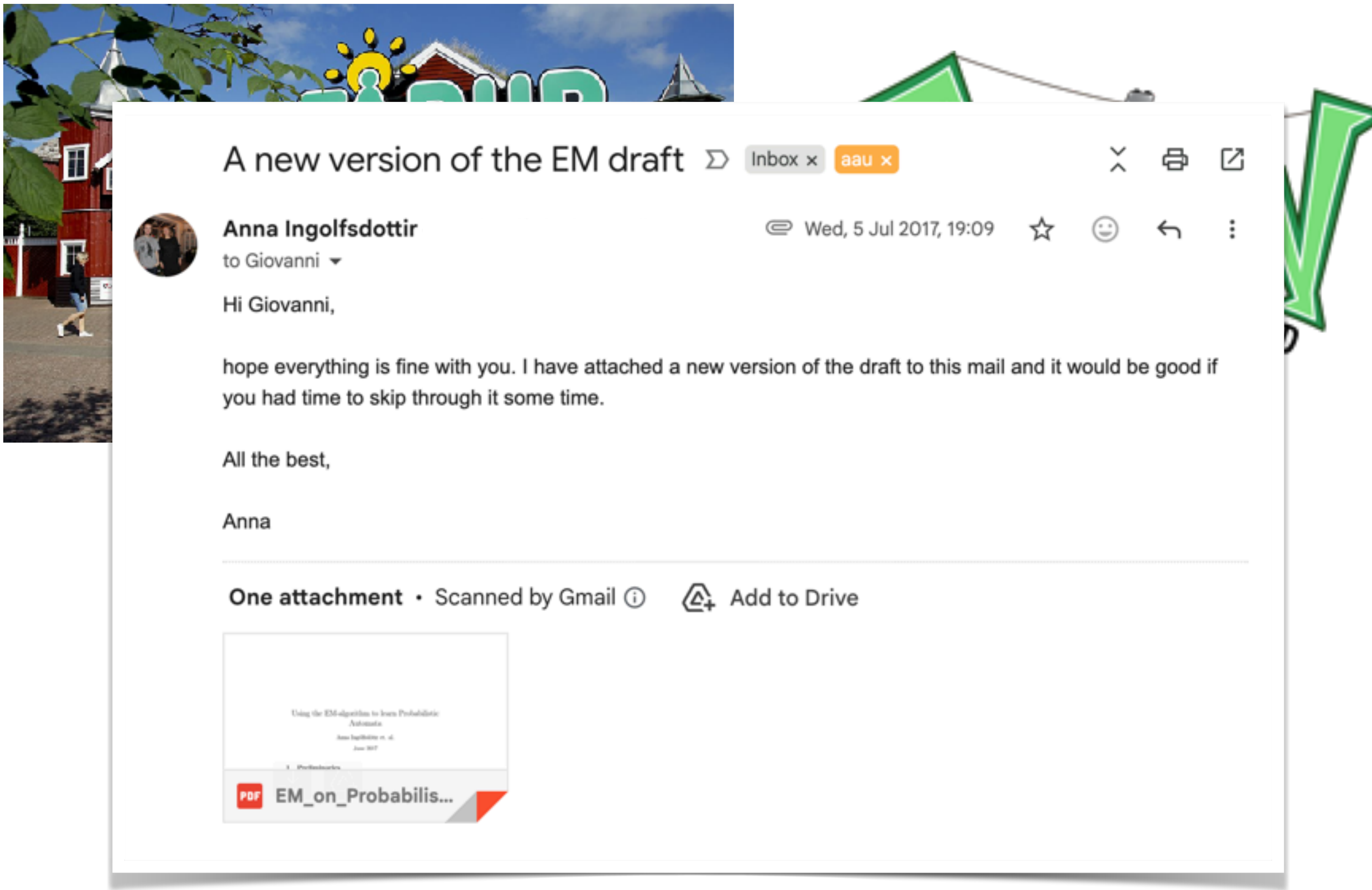
Anna Ingólfsdóttir, Kim G. Larsen, Raphaël Reynouard,
Sebastian Aaholm, Lars Emanuel Hansen, Daniel Runge Petersen

November 14, 2025 — Reykjavík, IS

Where all started

FARM FUN
DYREPARK & LEGELAND
9982 Ålbæk - Skagen

Where all started



A new version of the EM draft   Inbox ×   aau ×

Anna Ingolfsdottir
to Giovanni

Wed, 5 Jul 2017, 19:09

Hi Giovanni,

hope everything is fine with you. I have attached a new version of the draft to this mail and it would be good if you had time to skip through it some time.

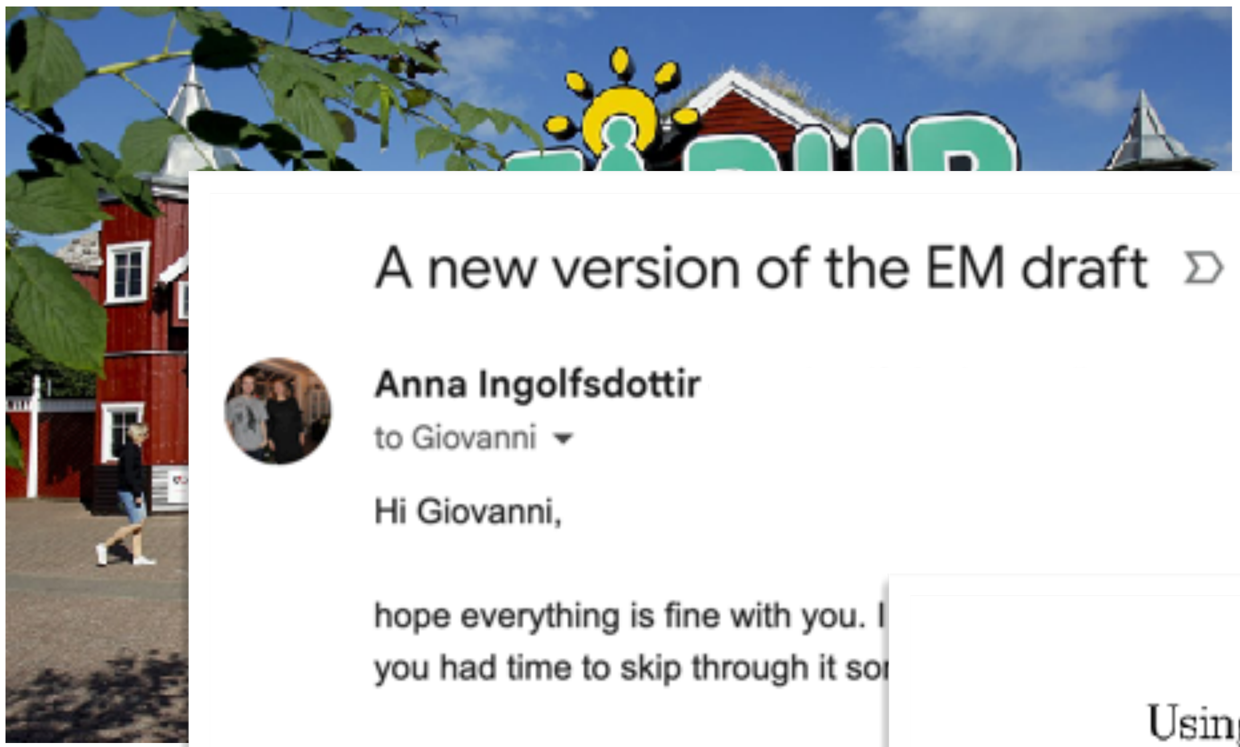
All the best,

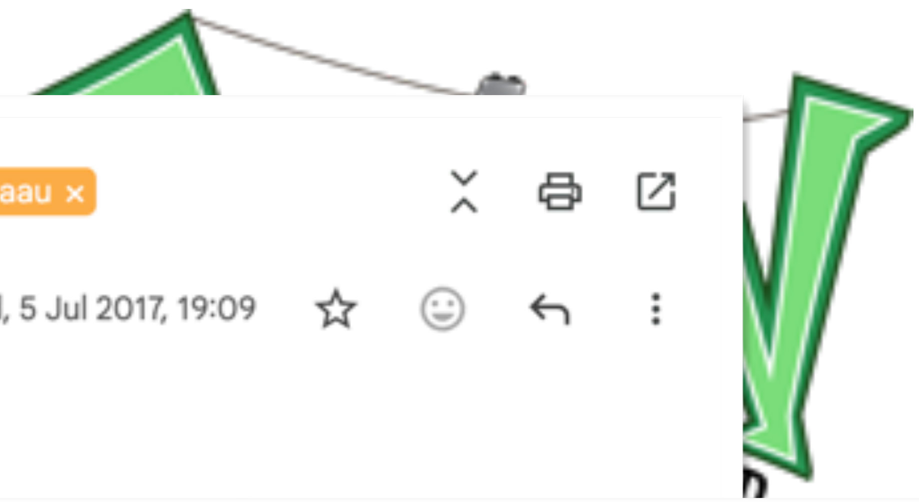
Anna

One attachment • Scanned by Gmail   Add to Drive

EM_on_Probabilis...

Where all started





A new version of the EM draft Inbox × aau ×

Anna Ingolfsdottir

to Giovanni

Wed, 5 Jul 2017, 19:09

Hi Giovanni,

hope everything is fine with you. I

you had time to skip through it so

All the best,

Anna

One attachment • Scanned

EM_on_Probabilis...

Using the EM-algorithm to learn Probabilistic Automata

Anna Ingólfsdótir et. al.

June 2017

1 Preliminaries

In this study we use *Generative Probabilistic Labelled Transition systems*, or *gpLTS*, as introduced in [] and defined below.

Definition 1.1 (*Generative Probabilistic Labelled Transition systems*). *A* gpLTS **P** *is a quadruple*

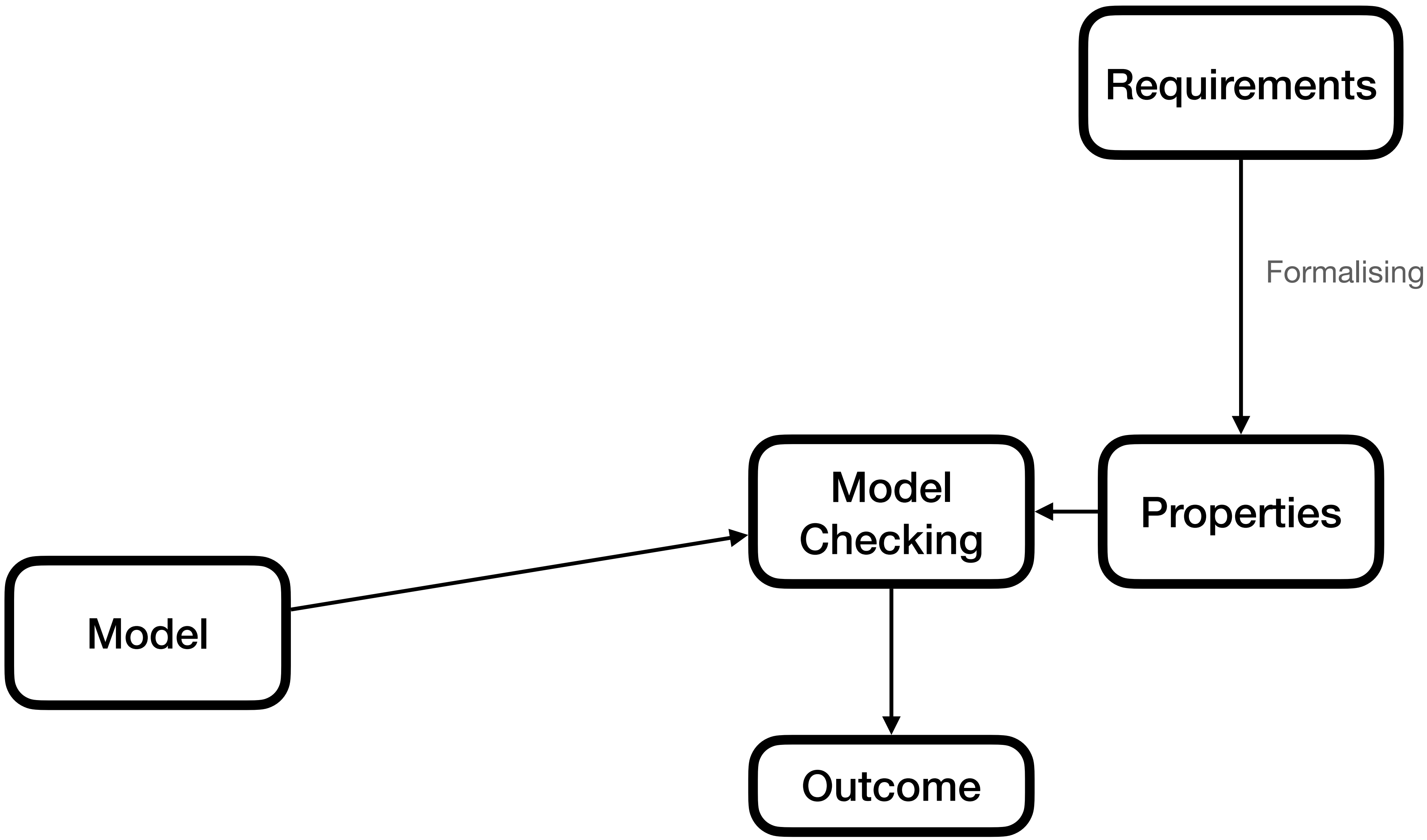
$$\mathbf{P} = (\mathcal{L}, \mathcal{S}, p, \mathbf{s}),$$

where

- $\mathcal{L}$ *is a set of* labels *(sometimes called* actions*) ranged over by* $a, b, \ldots$,
- $\mathcal{S}$ *is a set of states ranged over by* $s, t, \ldots$

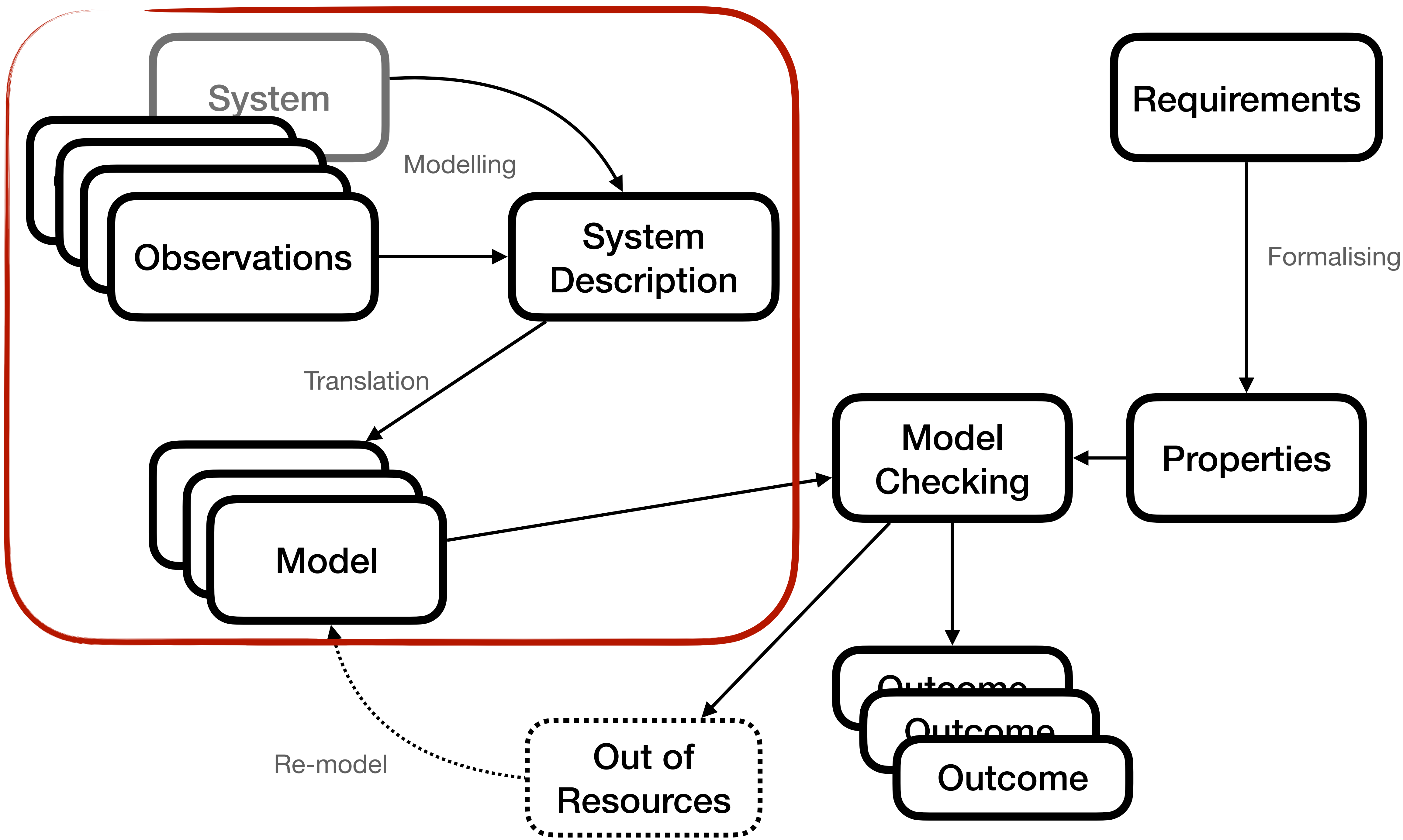
Model Checking Workflow

the typical textbook picture



Model Checking Workflow

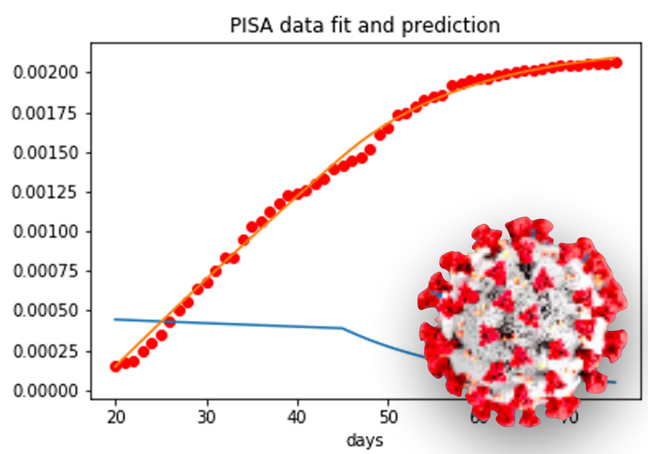
… a more realistic scenario



Case Study: covid spread in Toscany

Example of system modelling and analysis [Milazzo'21]

$$\frac{dS}{dt} = -\beta S I p(t)$$
$$\frac{dI}{dt} = \beta S I p(t) - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$



SciPy library

`optimize.curve_fit`

Infection coefficient

beta = 0.122

gamma = 0.127

Recovery coefficient

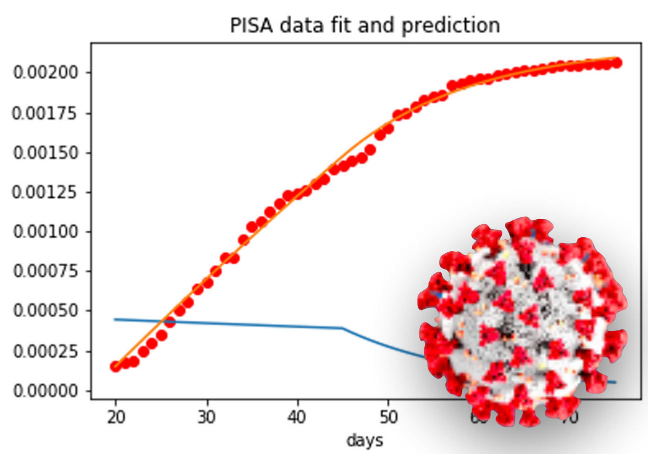
```
ctmc

const double beta = 0.122128; const double gamma = 0.127283;
const double plock = 0.472081; const int SIZE = 100000;

module SIR_Pisa

s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 -> beta*s*i*plock/SIZE : (s'=s-1)&(i'=i+1);
[] i>0 & r<SIZE -> gamma*i*plock : (i'=i-1)&(r'=r+1);

endmodule
```

Prism Model for Modified SIR

Case Study: covid spread in Toscany

Example of system modelling and analysis [Milazzo’21]

$$\frac{dS}{dt} = -\beta S I p(t)$$
$$\frac{dI}{dt} = \beta S I p(t) - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$



SciPy library
`optimize.curve_fit`

Infection coefficient

beta = 0.122
gamma = 0.127

Recovery coefficient

```
ctmc

const double beta = 0.122128; const double gamma = 0.127283;
const double plock = 0.47208                  SIZE = 100000;

module SIR_Pisa

s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 ->                                    )&(i'=i+1);
[] i>0 & r<SIZE -> gamma*i*plock : (i'=i-1)&(r'=r+1);

endmodule
```

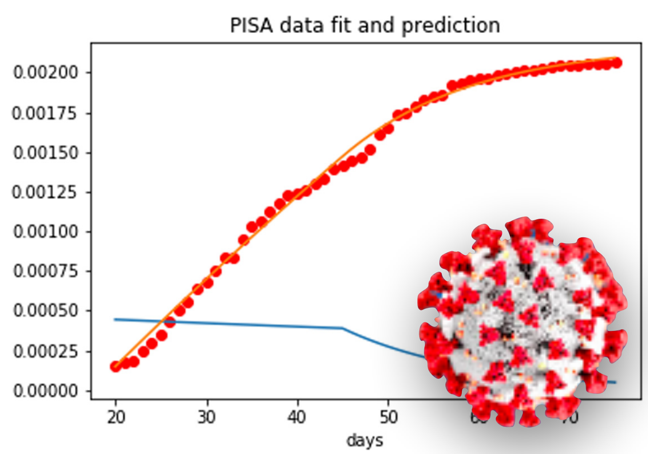
Too BIG

Prism Model for Modified SIR

Case Study: covid spread in Toscany

Example of system modelling and analysis [Milazzo'21]

$$\frac{dS}{dt} = -\beta S I p(t)$$
$$\frac{dI}{dt} = \beta S I p(t) - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$



Infection coefficient

beta = 0.122
gamma = 0.127

Recovery coefficient

SciPy library

```
ctmc

const double beta = 0.122128; const double gamma = 0.12728
const double plock = 0.47208

module SIR_Pisa

s : [0..SIZE] init 99936
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 ->
[] i>0 & r<SIZE -> gamma*i*plock : (i'=i-1)&(r'=r+1);

endmodule
```

Too BIG

Prism Model for Modified SIR

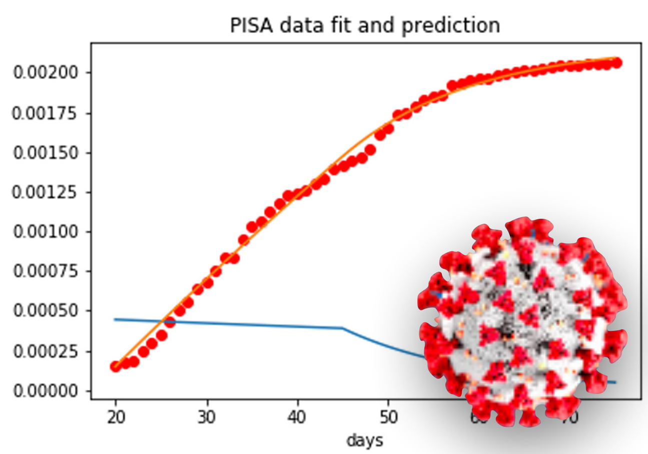
```
ctmc

const double beta = 0.122128; const double gamma = 0.127283;
const double plock = 0.472081; const int SIZE = 100000; const int BOUND = 500;

module SIR_Pisa

i : [0..BOUND] init 48;
r : [0..BOUND] init 16;

[] i>0 & i<BOUND -> beta*(SIZE-(i+r))*i*plock/SIZE : (i'=i+1);
[] i>0 & r<BOUND -> gamma*i*plock : (i'=i-1)&(r'=r+1);
[] i>0 & r=BOUND -> gamma*i*plock : (i'=i-1);

endmodule
```

Approximated Model for Modified SIR

Case Study: covid spread in Toscany

Example of system modelling and analysis [Milazzo'21]

$$\frac{dS}{dt} = -\beta S I p(t)$$
$$\frac{dI}{dt} = \beta S I p(t) - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$



Infection coefficient

beta = 0.122

gamma = 0.127

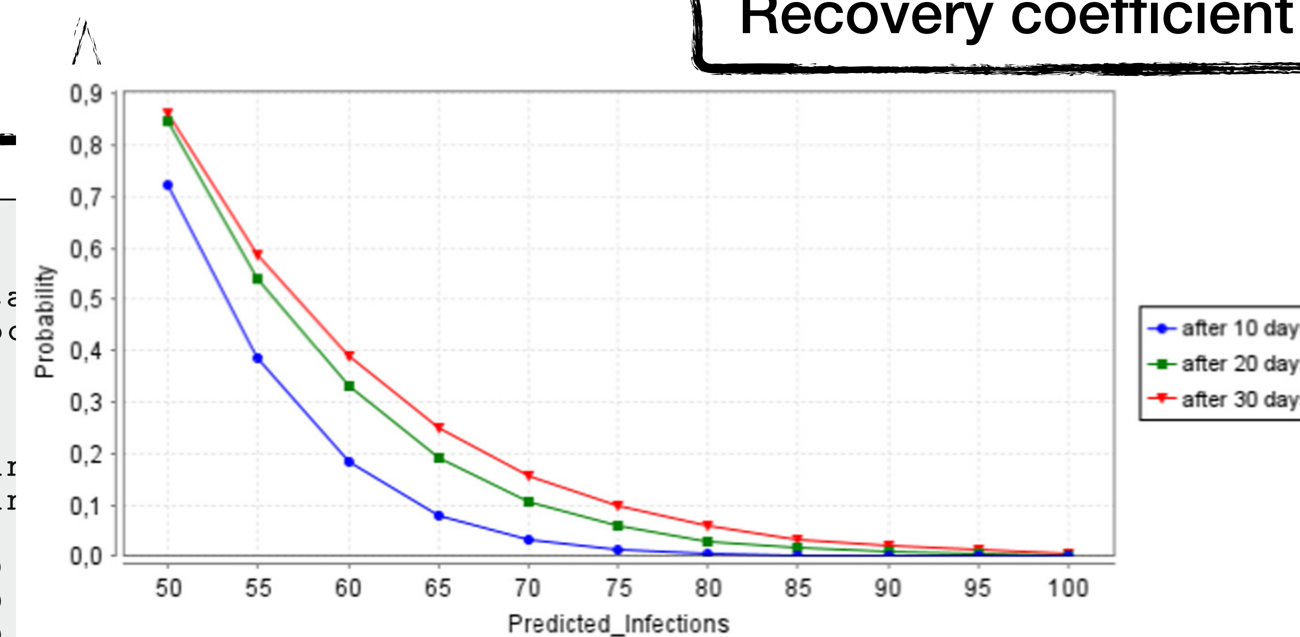
Recovery coefficient

```
ctmc

const double beta = 0.122128; const double gamma = 0.12728
const double plock = 0.472081 ... ;

module SIR_Pisa

s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 -> ... )&(i
[] i>0 & r<SIZE -> gamma*i*plock : (i'=i-1)&(r'=r+1);

endmodule
```

Too BIG

Prism Model for Modified SIR

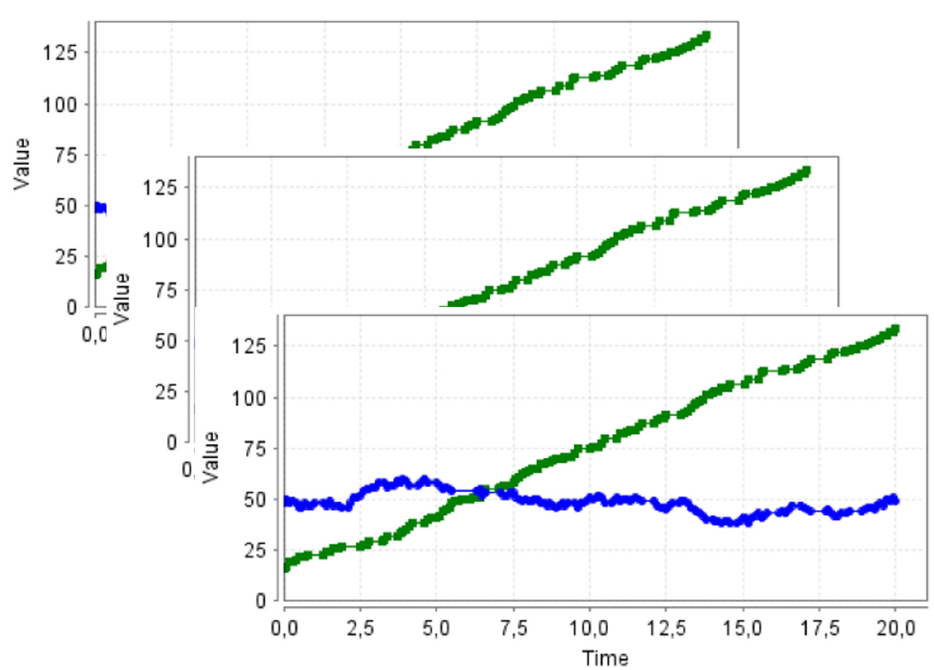
```
ctmc

const double beta
const double ploc

module SIR_Pisa

i : [0..BOUND] in
r : [0..BOUND] in

[] i>0 & i<BOUND
[] i>0 & r<BOUND
[] i>0 & r=BOUND

endmodule
```



Approximated Model for Modified SIR

Work directly on the modelling formalism

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module SIR
s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 →
   beta * s * i * plock/SIZE : (s'=s − 1)&(i'=i + 1);
[] i>0 & r<SIZE →
    gamma * i * plock : (i'=i − 1)&(r'=r + 1);
endmodule
```



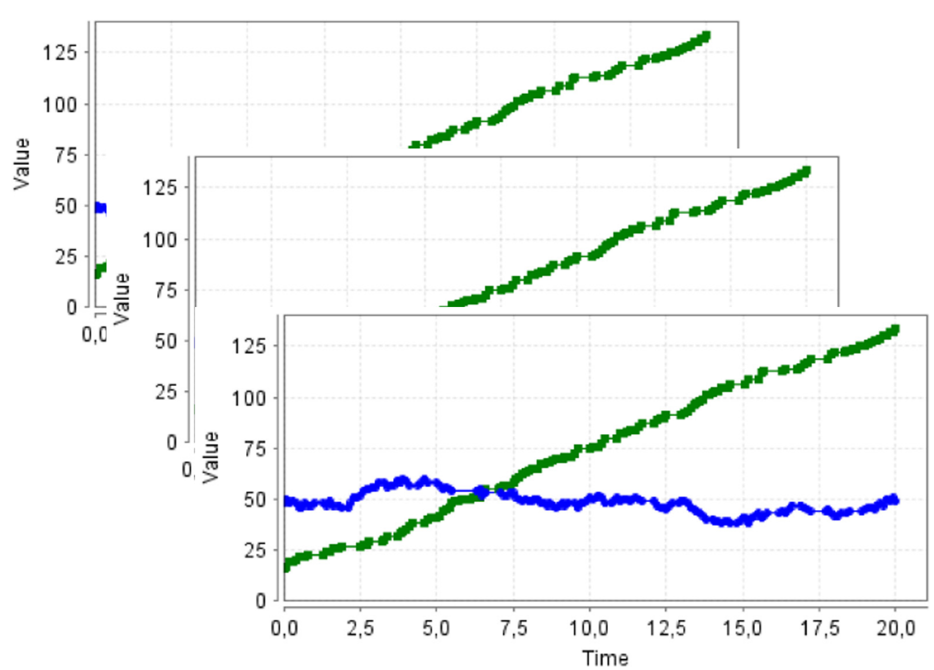
beta = 0.122
gamma = 0.127

Work directly on the modelling formalism

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module SIR
s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0  →
   beta * s * i * plock/SIZE : (s'=s − 1)&(i'=i + 1);
[] i>0 & r<SIZE  →
    gamma * i * plock : (i'=i − 1)&(r'=r + 1);
endmodule
```

Model template
(with parameters)

beta = 0.122
gamma = 0.127

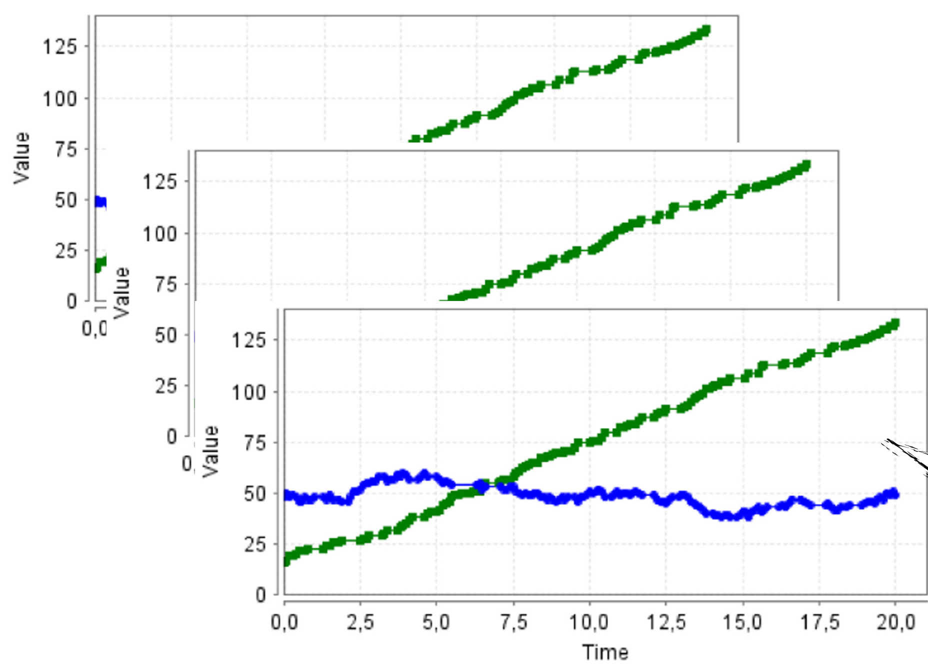


Work directly on the modelling formalism

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module SIR
s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0  →
   beta * s * i * plock/SIZE : (s′=s − 1)&(i′=i + 1);
[] i>0 & r<SIZE  →
    gamma * i * plock : (i′=i − 1)&(r′=r + 1);
endmodule
```

Model template
(with parameters)

beta = 0.122
gamma = 0.127



Observations
(partial observability)

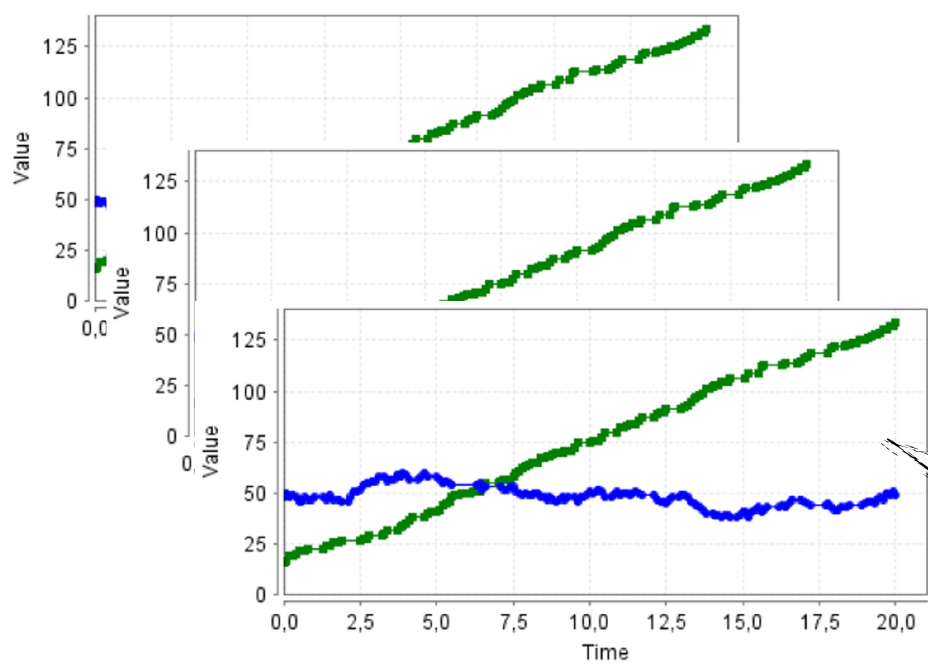
Work directly on the modelling formalism

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module SIR
s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0  →
   beta * s * i * plock/SIZE : (s'=s − 1)&(i'=i + 1);
[] i>0 & r<SIZE  →
    gamma * i * plock  : (i'=i − 1)&(r'=r + 1);
endmodule
```

Model template
(with parameters)

Parameters that fit
best the observation.

beta = 0.122
gamma = 0.127



Observations
(partial observability)

Work directly on the modelling formalism

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module SIR
s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 →
  beta * s * i * plock/SIZE : (s'=s − 1)&(i'=i + 1);
[] i>0 & r<SIZE →
   gamma * i * plock : (i'=i − 1)&(r'=r + 1);
endmodule
```

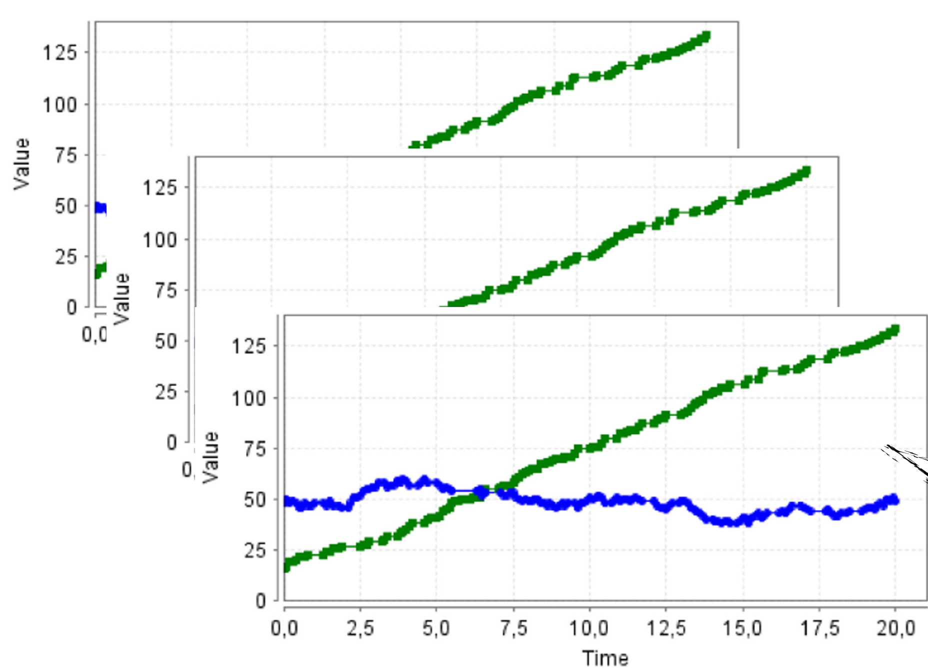
Model template
(with parameters)

Parameters that fit
best the observation.

beta = 0.122
gamma = 0.127

Maximum Likelihood Estimation

$$\mathcal{L}(\mathcal{P}(\mathbf{x})|\mathcal{O}) = \prod_{j=1}^{J} l(\mathbf{o}_j|\mathcal{P}(\mathbf{x}))$$



Observations
(partial observability)

Work directly on the modelling formalism

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
co
mo
s
i
r
[]
[]
en
```

Model template
(with parameters)

Parameters that fit
n.

Active Learning of Markov Decision Processes using Baum-Welch algorithm

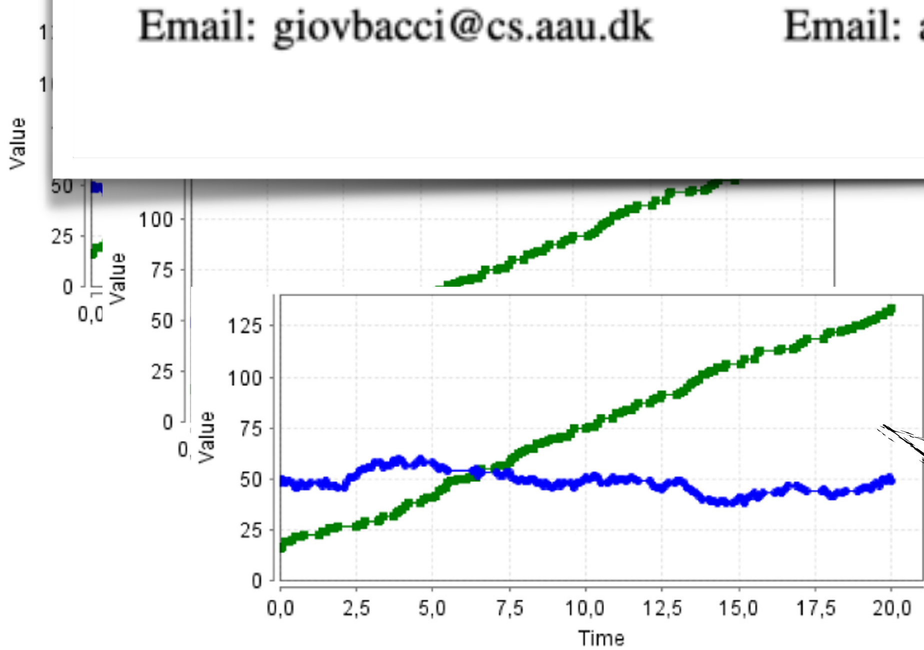
Giovanni Bacci
Dept. of Computer Science
Aalborg, Denmark
Email: giovbacci@cs.aau.dk

Anna Ingólfsdóttir
Dept. of Computer Science
Reykjavík, Iceland
Email: annai@ru.is

Kim G. Larsen
Dept. of Computer Science
Aalborg, Denmark
Email: kgl@cs.aau.dk

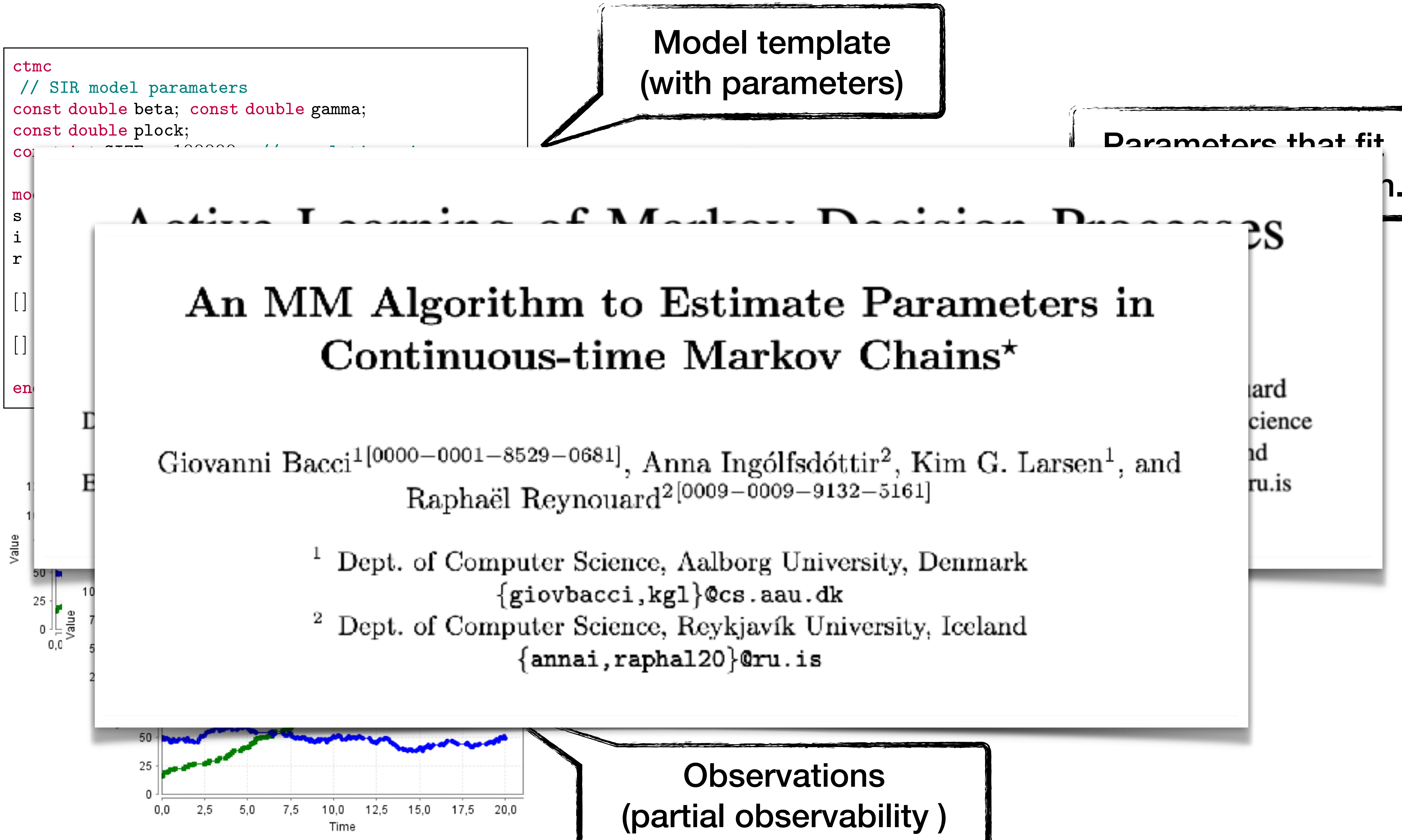
Raphaël Reynouard
Dept. of Computer Science
Reykjavík, Iceland
Email: raphal20@ru.is

$$\mathcal{L}(\mathcal{P}(\mathbf{x})|\mathcal{O}) = \prod_{j=1} l(\mathbf{o}_j|\mathcal{P}(\mathbf{x}))$$



Observations
(partial observability)

Work directly on the modelling formalism


An MM Algorithm to Estimate Parameters in Continuous-time Markov Chains*

Giovanni Bacci[1][0000−0001−8529−0681], Anna Ingólfsdóttir[2], Kim G. Larsen[1], and Raphaël Reynouard[2][0009−0009−9132−5161]

[1] Dept. of Computer Science, Aalborg University, Denmark
{giovbacci,kgl}@cs.aau.dk
[2] Dept. of Computer Science, Reykjavík University, Iceland
{annai,raphal20}@ru.is

Work directly on the modelling formalism

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
co
mo
s
i
r
[]
```

Model template
(with parameters)

Parameters that fit
n.

Active Learning of Markov Decision Processes

ard
cience
nd
ru.is

An MM Algorithm to Estimate Parameters in

Jajapy: a learning library for stochastic models *

Raphaël Reynouard[1] [0009−0009−9132−5161], Anna Ingólfsdóttir[1], and Giovanni Bacci[2] [0000−0001−8529−0681]

[1] Reykjavík University, Iceland
[2] Aalborg University, Denmark

QEST Artifact Evaluation 2023 Accepted

0,0 2,5 5,0 7,5 10,0 12,5 15,0 17,5 20,0
Time

(partial observability)

jajapy

A library to learn Markov models

Raphaël Reynouard

- The library contains a number of algorithms to learn
  - DTMCs, MDPs, CTMCs, HMMs, and GoHMMs
- Interoperable with STORM via StormPy
- Compatible with PRISM via import/export of PRISM models

Check it out at https://pypi.org/project/jajapy/

The End

The End

…not quite

Scalability is still an issue

Problem: Jajapy uses explicit state-space model representation

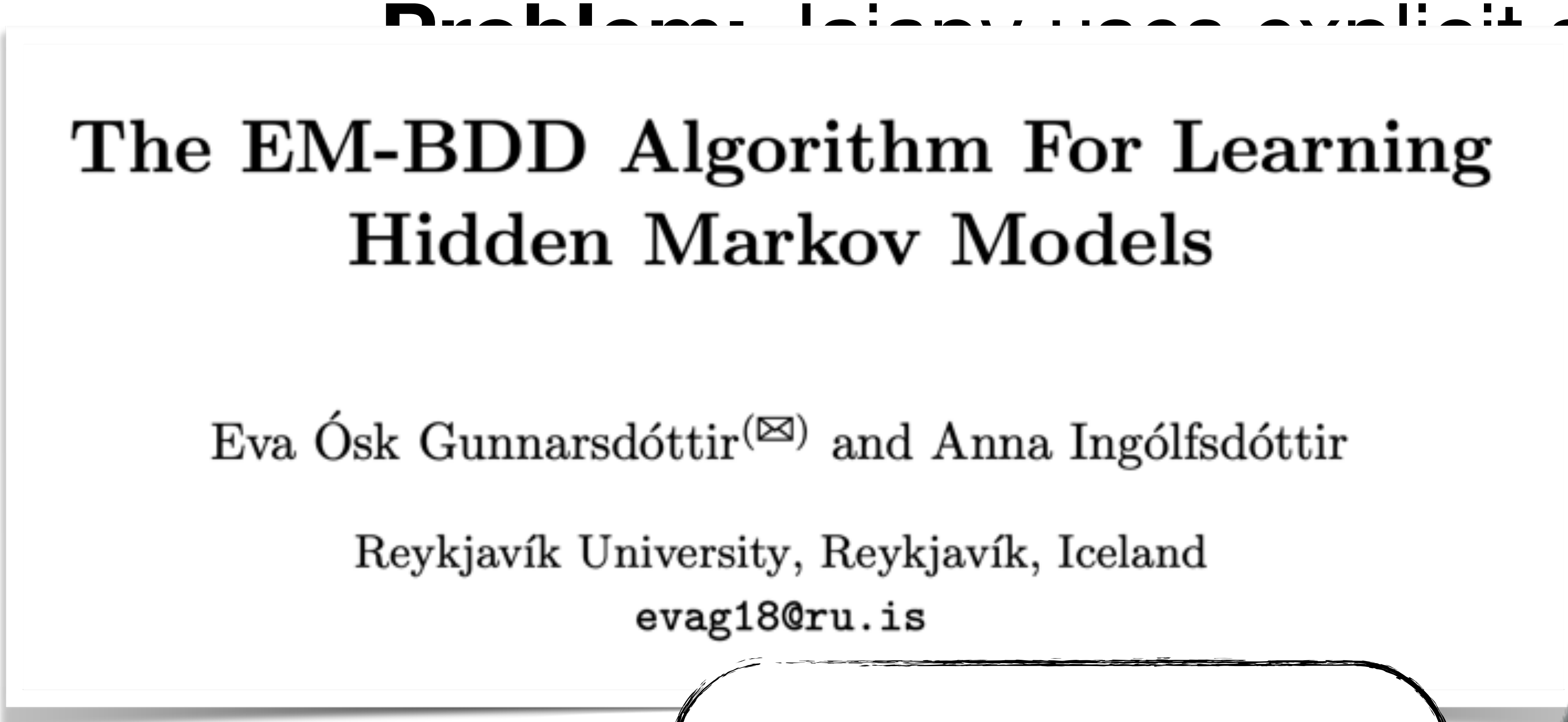
It does not scale to realistic models

Scalability is still an issue

state-

n

odels



The EM-BDD Algorithm For Learning Hidden Markov Models

Eva Ósk Gunnarsdóttir(✉) and Anna Ingólfsdóttir

Reykjavík University, Reykjavík, Iceland

evag18@ru.is

Markov Chains

$$\mathscr{M} = (S, P \in [0,1]^{n \times n}, \pi \in [0,1]^n, \ell \in AP^n)$$

States $\{1 \ldots n\}$

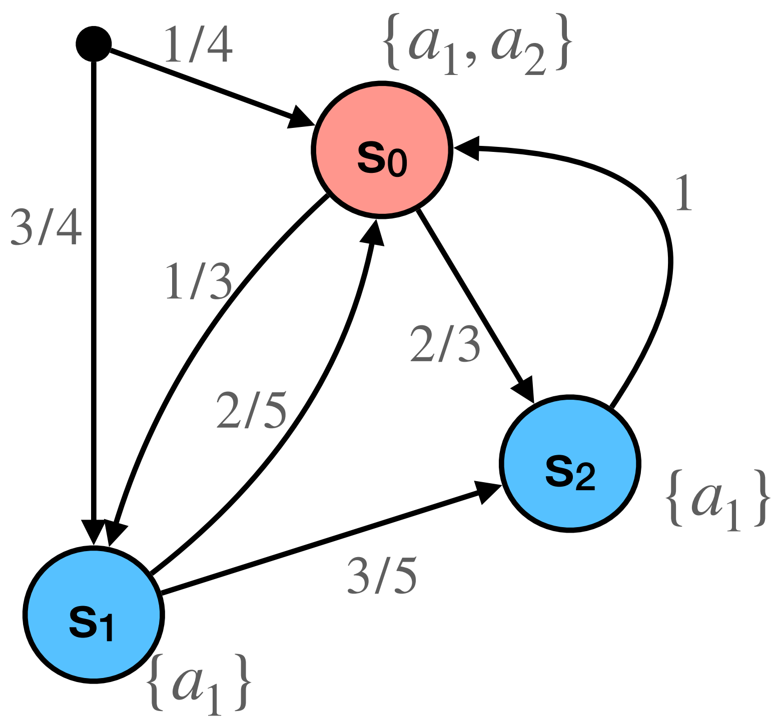
Probabilistic transition matrix

$p_{i,j} = P[X_{t+1} = s_j | X_t = s_i]$

Initial distribution

$\pi_i = P[X_1 = s_i]$

State labelling vector



Markov Chains

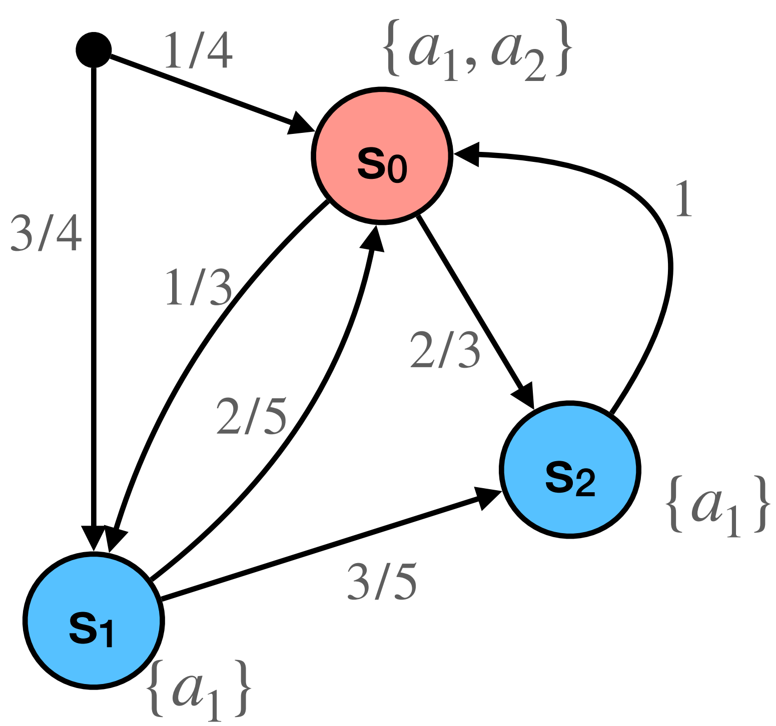
$$\mathscr{M} = (S, P \in [0,1]^{n\times n}, \pi \in [0,1]^n, \ell \in AP^n)$$

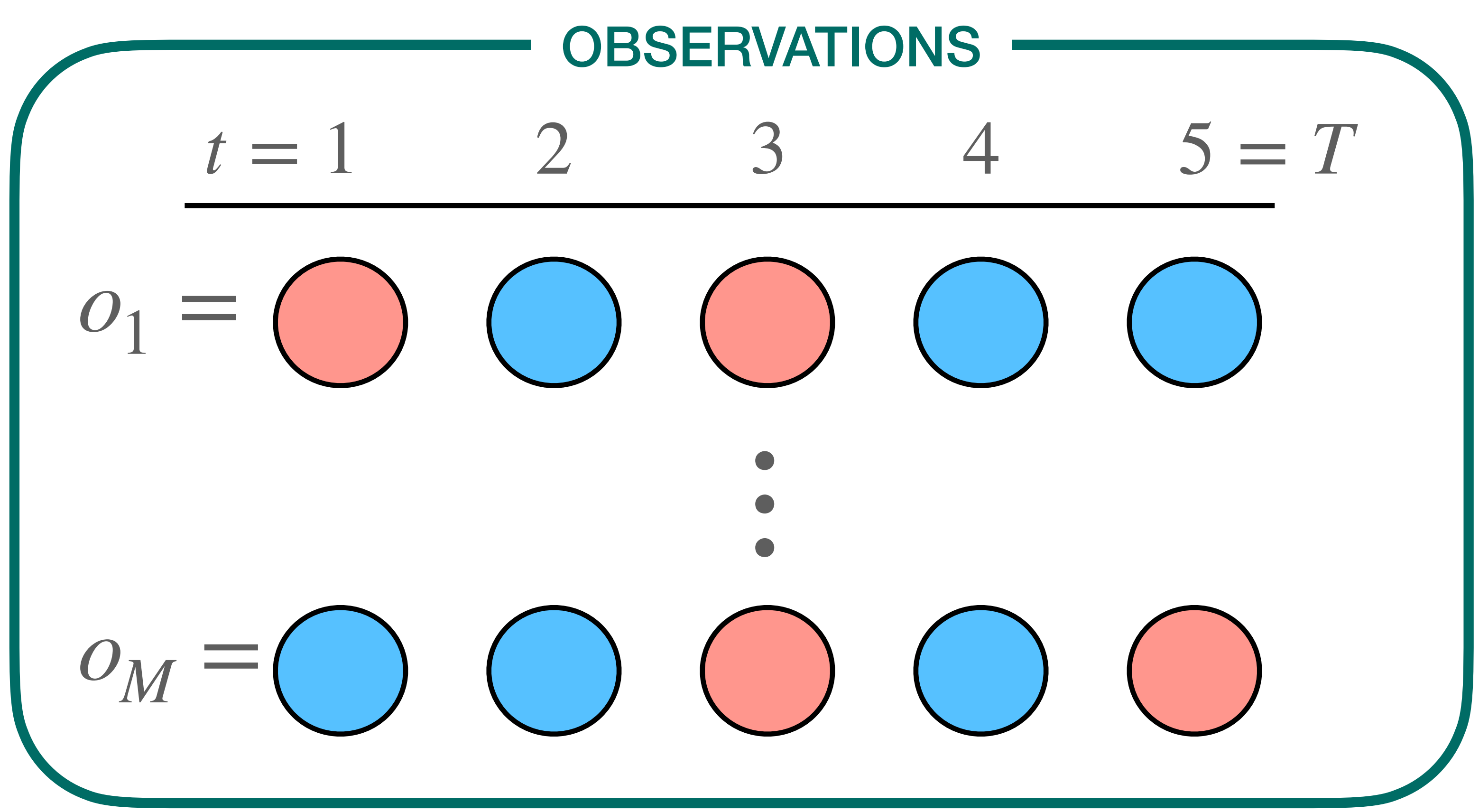
States $\{1 \dots n\}$

Probabilistic transition matrix
$p_{i,j} = P[X_{t+1} = s_j \,|\, X_t = s_i]$

Initial distribution
$\pi_i = P[X_1 = s_i]$

State labelling vector





Markov Chains

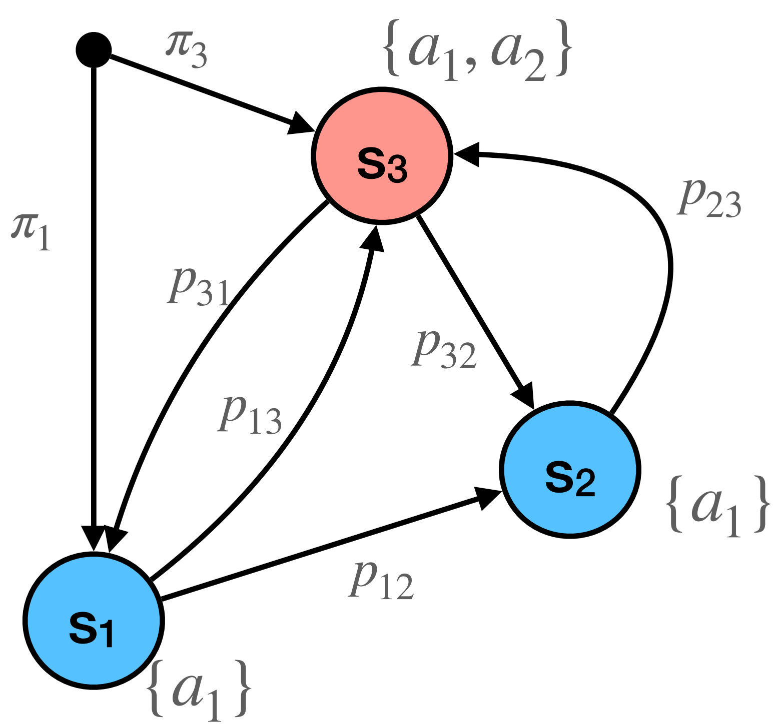
$$\mathscr{M} = (S, P \in [0,1]^{n \times n}, \pi \in [0,1]^n, \ell \in AP^n)$$

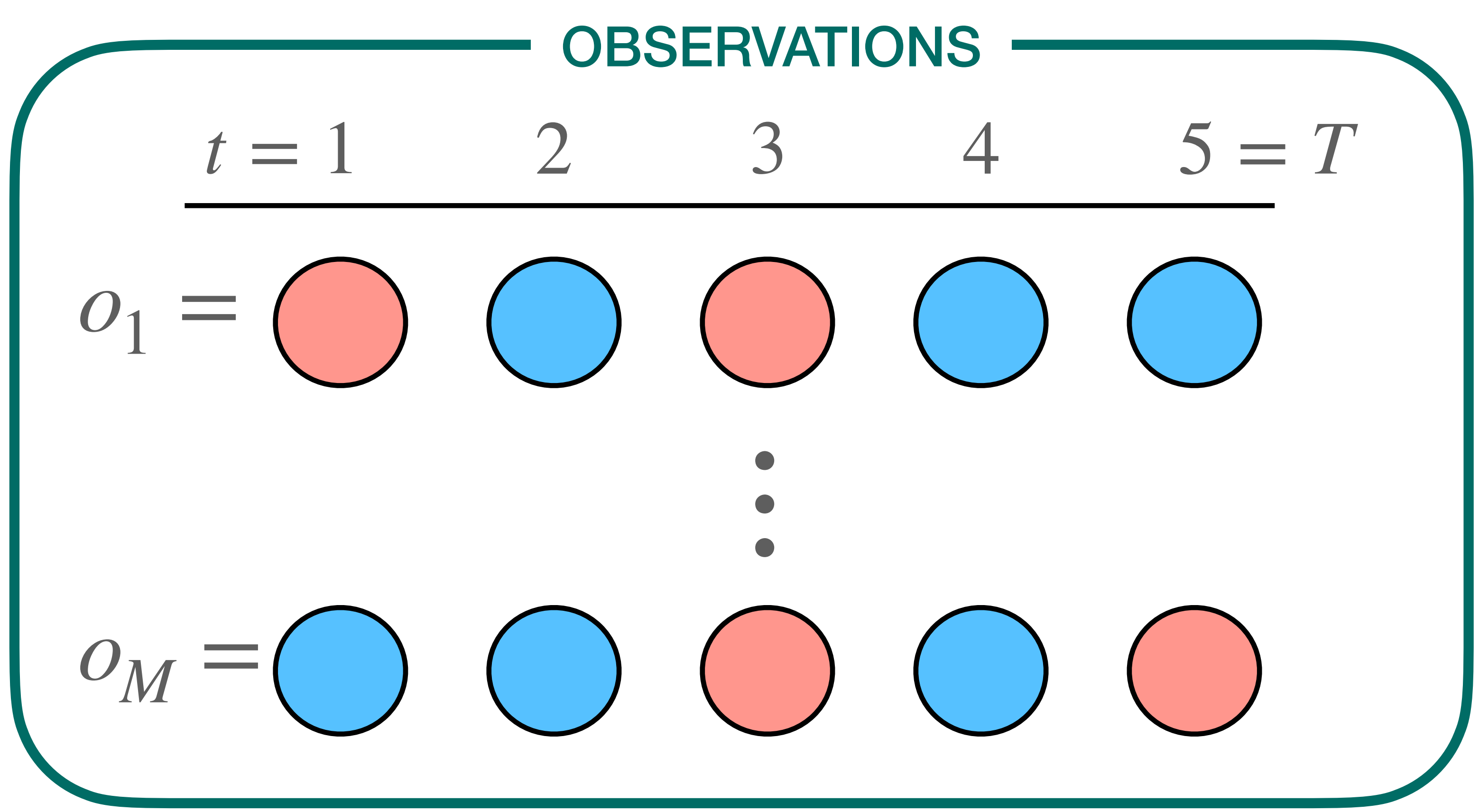
States $\{1 \ldots n\}$

Probabilistic transition matrix
$p_{i,j} = P[X_{t+1} = s_j \mid X_t = s_i]$

Initial distribution
$\pi_i = P[X_1 = s_i]$

State labelling vector





Markov Chains

$$\mathscr{M} = (S, P \in [0,1]^{n \times n}, \pi \in [0,1]^n, \ell \in AP^n)$$

States $\{1 \ldots n\}$

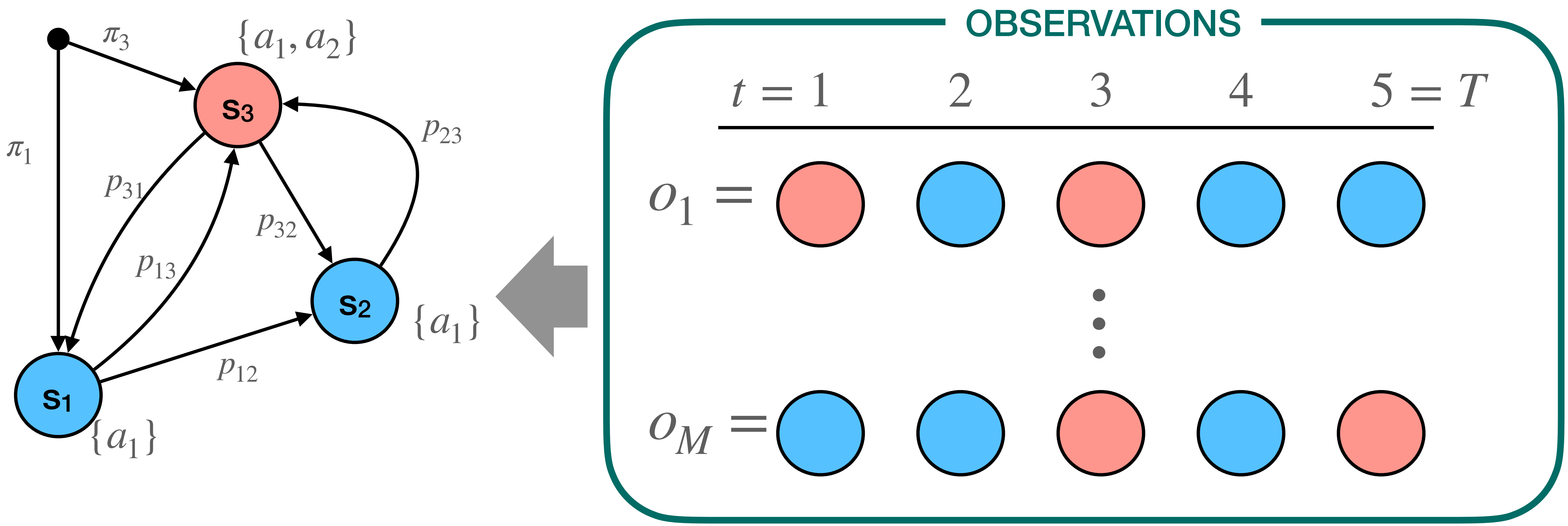
Probabilistic transition matrix

$p_{i,j} = P[X_{t+1} = s_j \,|\, X_t = s_i]$

Initial distribution

$\pi_i = P[X_1 = s_i]$

State labelling vector



Baum Welch Algorithm

in a nutsnell

Observation $\mathbf{o} = o_1 \cdots o_T$

$$\mathscr{M}_h = (S, P, \pi, \ell)$$

$\omega_i(t) = 1$ iff $o_t \in \ell(s_i)$

E-Step

FORWARD-ALGORITHM

1 $\boldsymbol{\alpha}(1) = \boldsymbol{\omega}(1) \circ \boldsymbol{\pi}$
2 **for** $t = 2$ **to** T
3 $\quad \boldsymbol{\alpha}(t) = \boldsymbol{\omega}(t) \circ \left(P^\top \boldsymbol{\alpha}(t-1)\right)$

BACKWARD-ALGORITHM

1 $\boldsymbol{\beta}(T) = \mathbf{1}$
2 **for** $t = T-1$ **to** 1
3 $\quad \boldsymbol{\beta}(t) = P\left(\boldsymbol{\beta}(t+1) \circ \boldsymbol{\omega}(t+1)\right)$

$$\boldsymbol{\gamma}(t) = (\boldsymbol{\alpha}(t) \circ \boldsymbol{\beta}(t)) / P[\mathbf{o}|\mathcal{M}] \qquad t = 1 \ldots T$$

$$\boldsymbol{\xi}(t) = (P[\mathbf{o}|\mathcal{M}] \cdot P) \circ \left(\boldsymbol{\alpha}(t) \otimes (\boldsymbol{\beta}(t+1) \circ \boldsymbol{\omega}(t+1))^\top\right) \qquad t = 1 \ldots T-1$$

M-Step

$$\hat{\boldsymbol{\pi}} = \boldsymbol{\gamma}(1)$$ Improved initial distribution

$$\hat{P} = (\mathbf{1} \oslash \boldsymbol{\gamma}) \bullet \boldsymbol{\xi}$$ Improved probability matrix

$\sum_{t=1}^{T} \boldsymbol{\gamma}(t)$

$\sum_{t=1}^{T} \boldsymbol{\xi}(t)$

From explicit to symbolic

Symbolic data structures

- Usually based on binary decision diagrams (**BDDs**) or variants
- Avoid explicit enumeration of data by exploiting regularity
- Potentially very compact storage (but not always)

	Explicit	**Symbolic**
Set of states	Bit vectors	BDDs
Real-valued vectors	Arrays of reals*	ADDs
Real-valued matrices	Sparse matrices	ADDs

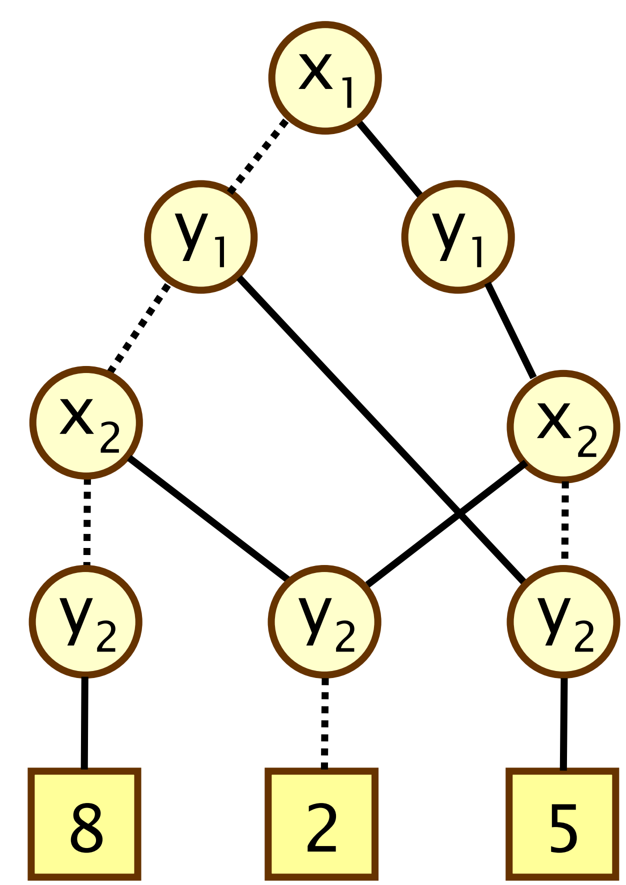
Algebraic decision diagrams (ADDs)

- Extension of BDDs to represent real-valued functions
- Like BDDs, and ADD is associated with n Boolean variables
- An ADD M represent a function $f_M \colon \{0,1\}^n \to \mathbb{R}$

$$M = \begin{bmatrix} 0 & 8 & 0 & 5 \\ 2 & 0 & 0 & 5 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 2 & 0 \end{bmatrix}$$

Entry in M	x_1	x_2	y_1	y_2	$x_1y_1x_2y_2$	f_M
(0,1) = 8	0	0	0	1	0001	8
(1,0) = 2	0	1	0	0	0010	2
(0,3) = 5	0	0	1	1	0101	5
(1,3) = 5	0	1	1	1	0111	5
(2,3) = 5	1	0	1	1	1101	5
(3,2) = 2	1	1	1	0	1110	2

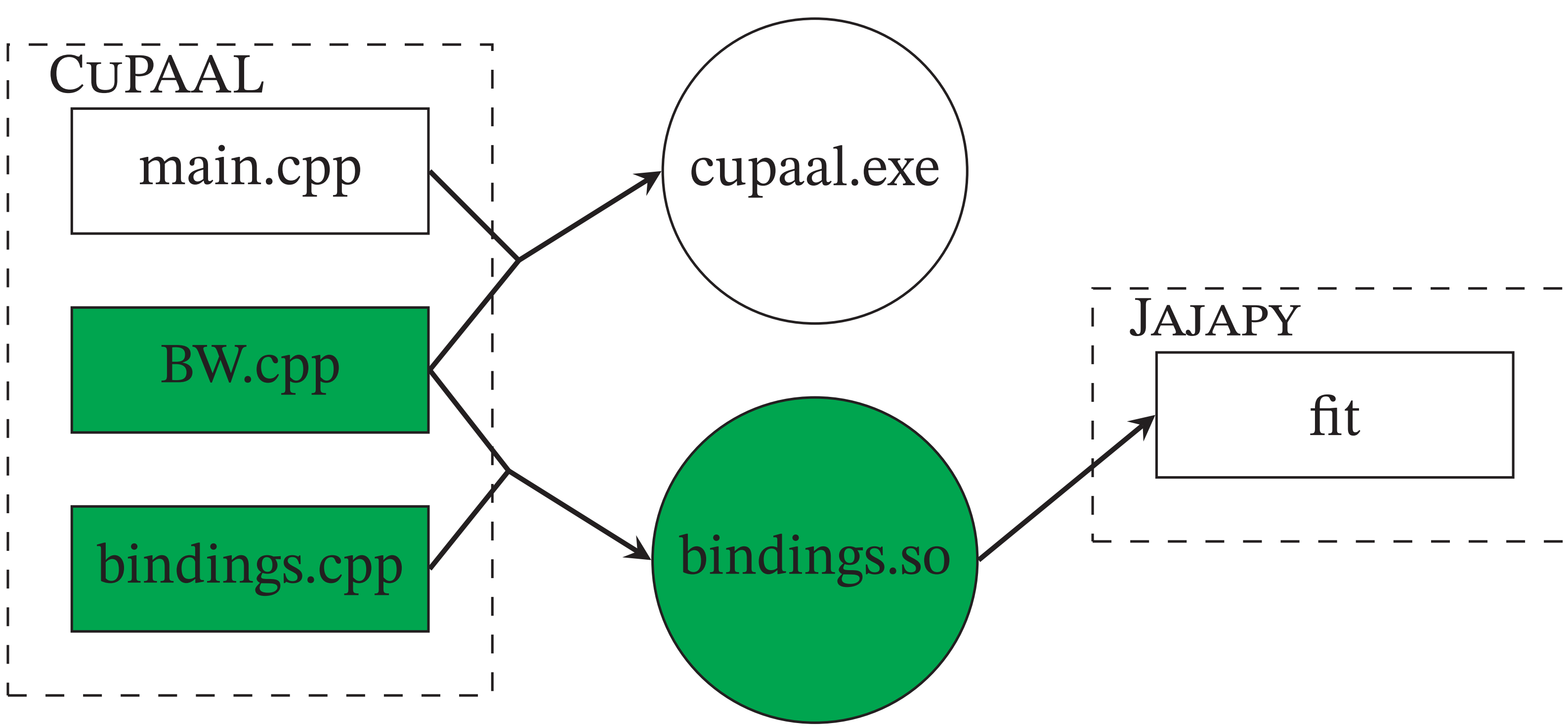
ADD for f_M



Figures stolen from https://www.prismmodelchecker.org/lectures/biss07/

Introducing CuPAAL

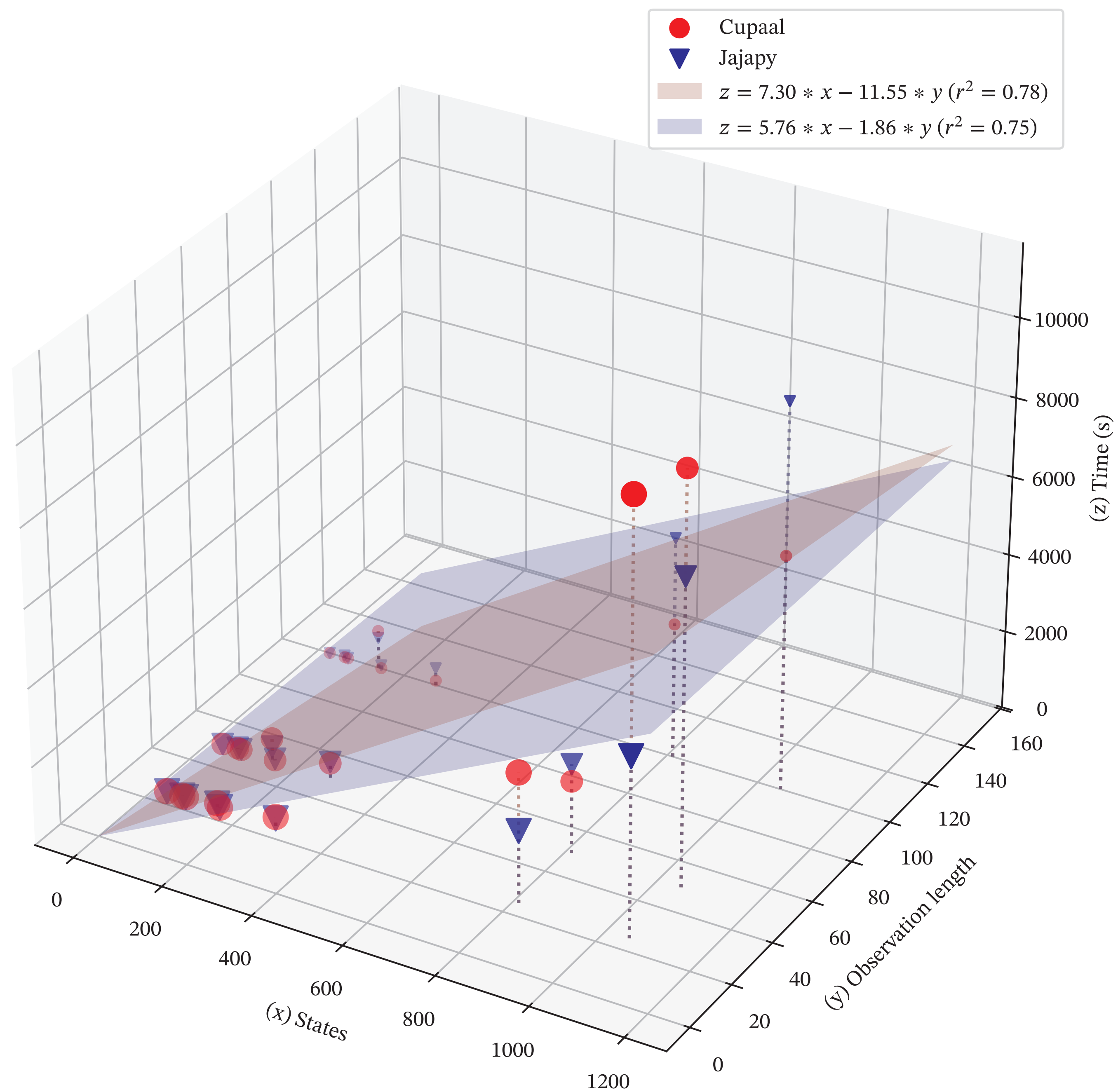
- We implemented the BW algorithm using ADDs
- The implementation uses the CUDD library extending it with
  - Hadamard product
  - A specialised version of the Kronecker product
- It integrates with Jajapy



Comparison with Jajapy

model	states	length	jajapy (s)	cupaal (s)
3.2	26	25	1.38	0.26
3.2	26	50	1.95	0.14
3.2	26	100	4.09	0.23
3.3	69	25	7.95	2.46
3.3	69	50	11.20	1.59
3.3	69	100	19.65	1.75
3.4	147	25	27.10	8.54
3.4	147	50	42.57	9.20
3.4	147	100	84.02	9.90
4.2	61	25	15.68	11.18
4.2	61	50	24.87	13.56
4.2	61	100	52.11	11.24
4.3	274	25	194.88	231.28
4.3	274	50	414.30	379.21
4.3	274	100	447.83	117.78
4.4	812	25	1846.68	3324.83
4.4	812	50	2290.28	1848.44
4.4	812	100	5652.14	3447.56
5.2	141	25	95.59	104.71
5.2	141	50	342.05	553.66
5.2	141	100	798.73	982.97
5.3	1050	25	4586.86	10906.91
5.3	1050	50	7791.95	10405.75
5.3	1050	100	9821.74	5992.51

Leader sync [Itai & Rodeh]
with N processes and K range of probabilistic choice

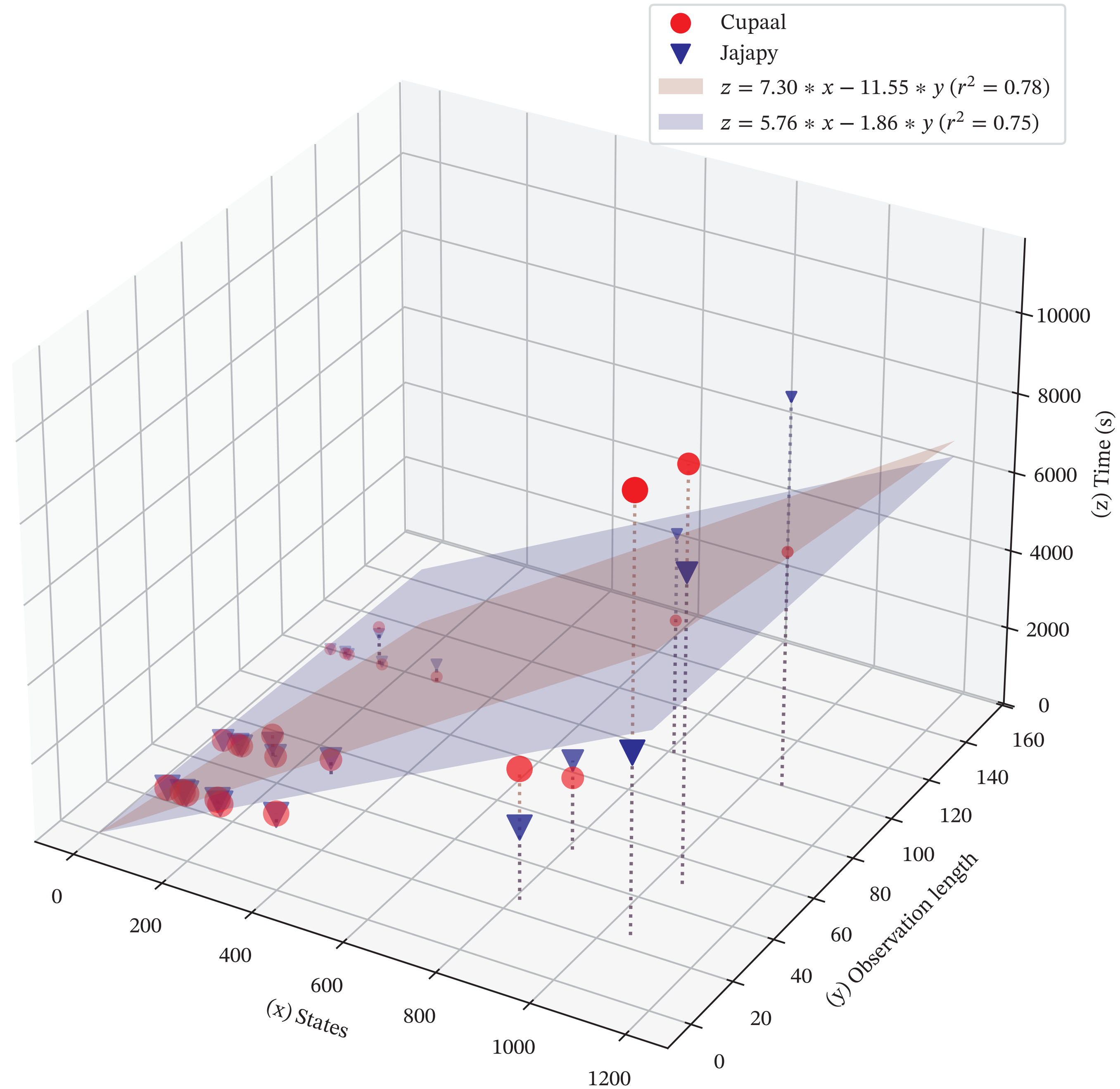


https://github.com/AAU-Dat/P10-Thesis/blob/main/experiments/original-models/dtmc/leader_sync.4-3.v1.prism

Comparison with Jajapy

model	states	length	jajapy (s)	cupaal (s)
3.2	26	25	1.38	0.26
3.2	26	50	1.95	0.14
3.2	26	100	4.09	0.23
3.3	69	25	7.95	2.46
3.3	69	50	11.20	1.59
3.3	69	100	19.65	1.75
3.4	147	25	27.10	8.54
3.4	147	50	42.57	9.20
3.4	147	100	84.02	9.90
4.2	61	25	15.68	11.18
4.2	61	50	24.87	13.56
4.2	61	100	52.11	11.24
4.3	274	25	194.88	231.28
4.3	274	50	414.30	379.21
4.3	274	100	447.83	117.78
4.4	812	25	1846.68	3324.83
4.4	812	50	2290.28	1848.44
4.4	812	100	5652.14	3447.56
5.2	141	25	95.59	104.71
5.2	141	50	342.05	553.66
5.2	141	100	798.73	982.97
5.3	1050	25	4586.86	10906.91
5.3	1050	50	7791.95	10405.75
5.3	1050	100	9821.74	5992.51

Leader sync [Itai & Rodeh]
with N processes and K range of probabilistic choice

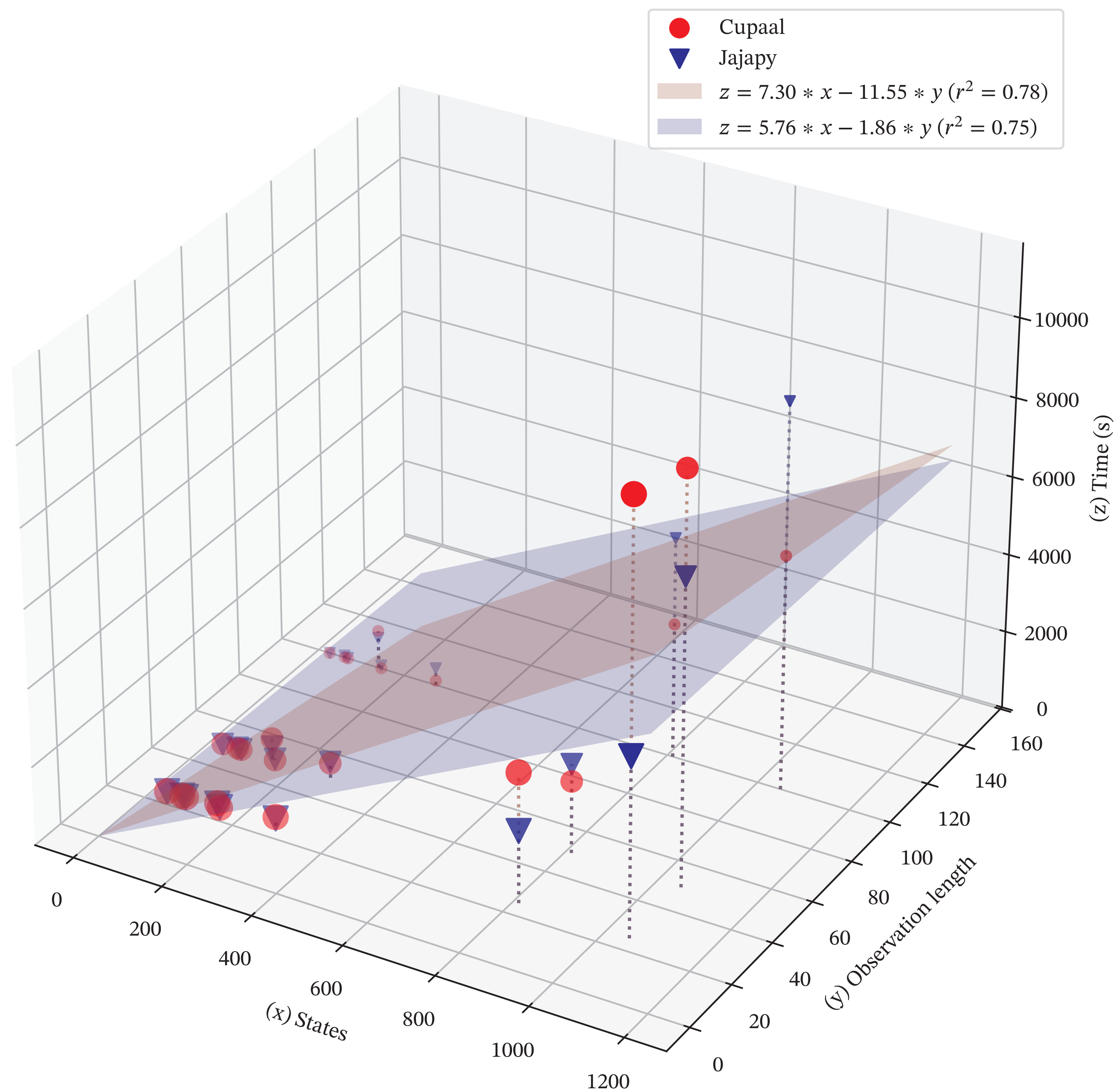


https://github.com/AAU-Dat/P10-Thesis/blob/main/experiments/original-models/dtmc/leader_sync.4-3.v1.prism

Comparison with Jajapy

model	states	length	jajapy (s)	cupaal (s)
3.2	26	25	1.38	0.26
3.2	26	50	1.95	0.14
3.2	26	100	4.09	0.23
3.3	69	25	7.95	2.46
3.3	69	50	11.20	1.59
3.3	69	100	19.65	1.75
3.4	147	25	27.10	8.54
3.4	147	50	42.57	9.20
3.4	147	100	84.02	9.90
4.2	61	25	15.68	11.18
4.2	61	50	24.87	13.56
4.2	61	100	52.11	11.24
4.3	274	25	194.88	231.28
4.3	274	50	414.30	379.21
4.3	274	100	447.83	117.78
4.4	812	25	1846.68	3324.83
4.4	812	50	2290.28	1848.44
4.4	812	100	5652.14	3447.56
5.2	141	25	95.59	104.71
5.2	141	50	342.05	553.66
5.2	141	100	798.73	982.97
5.3	1050	25	4586.86	10906.91
5.3	1050	50	7791.95	10405.75
5.3	1050	100	9821.74	5992.51

Leader sync [Itai & Rodeh]
with N processes and K range of probabilistic choice

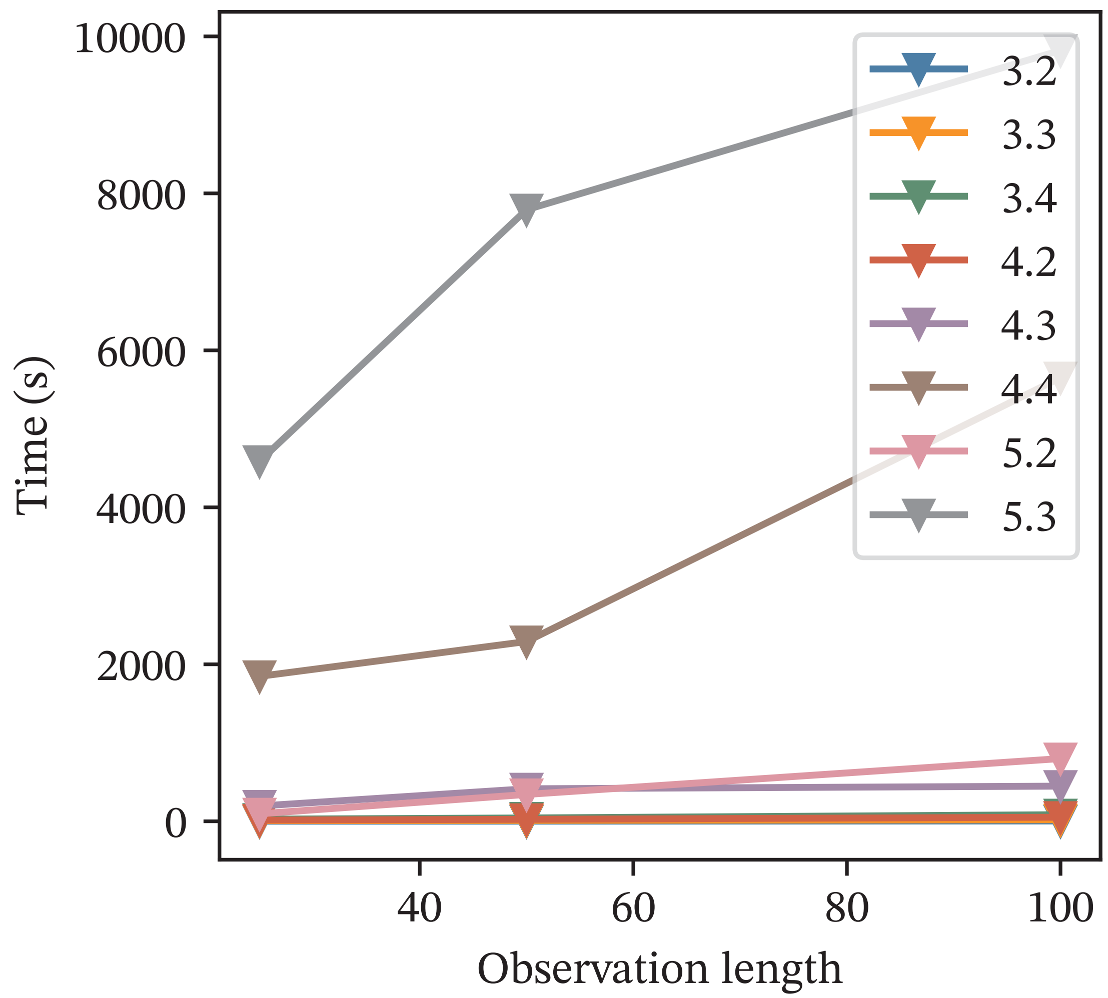


https://github.com/AAU-Dat/P10-Thesis/blob/main/experiments/original-models/dtmc/leader_sync.4-3.v1.prism

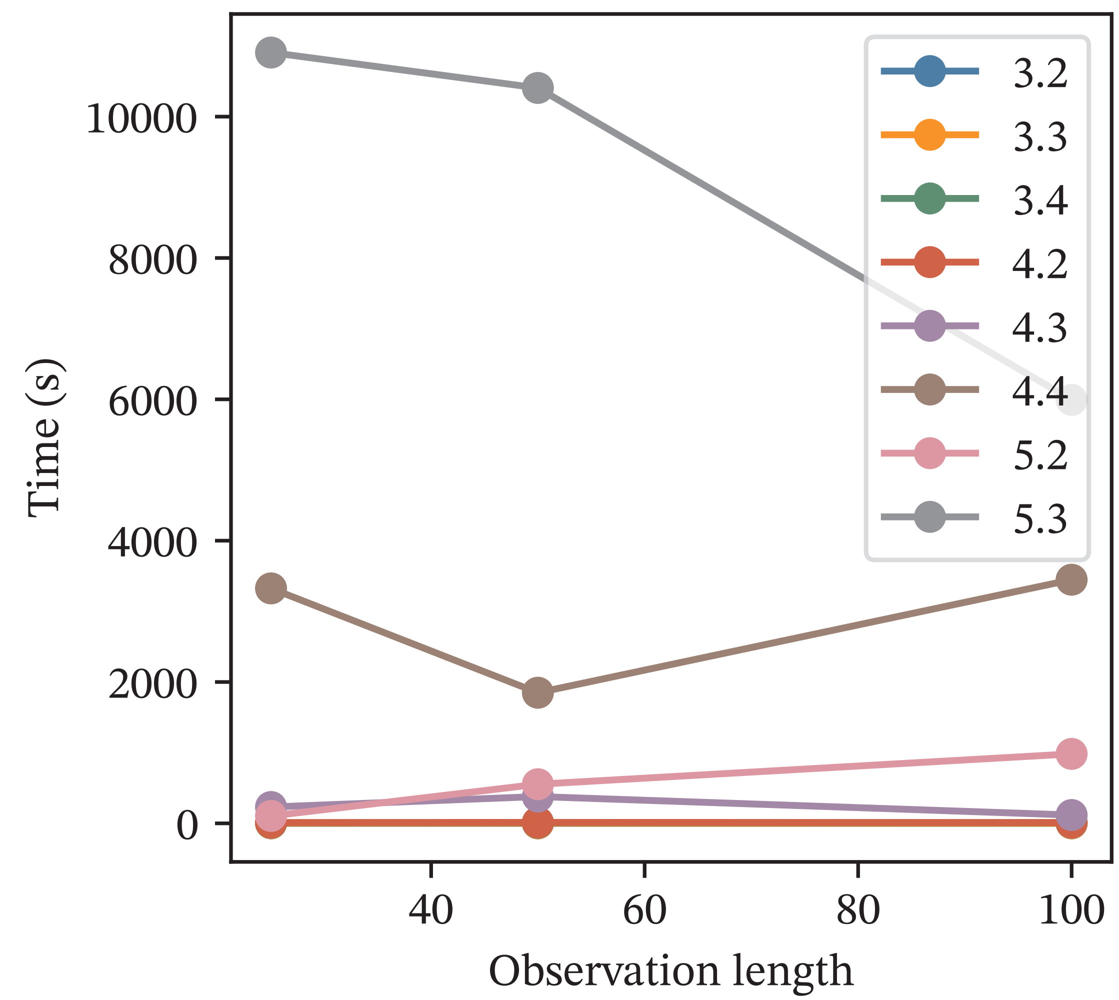
Comparison with Jajapy

a closer look

JajaPy



CuPAAL

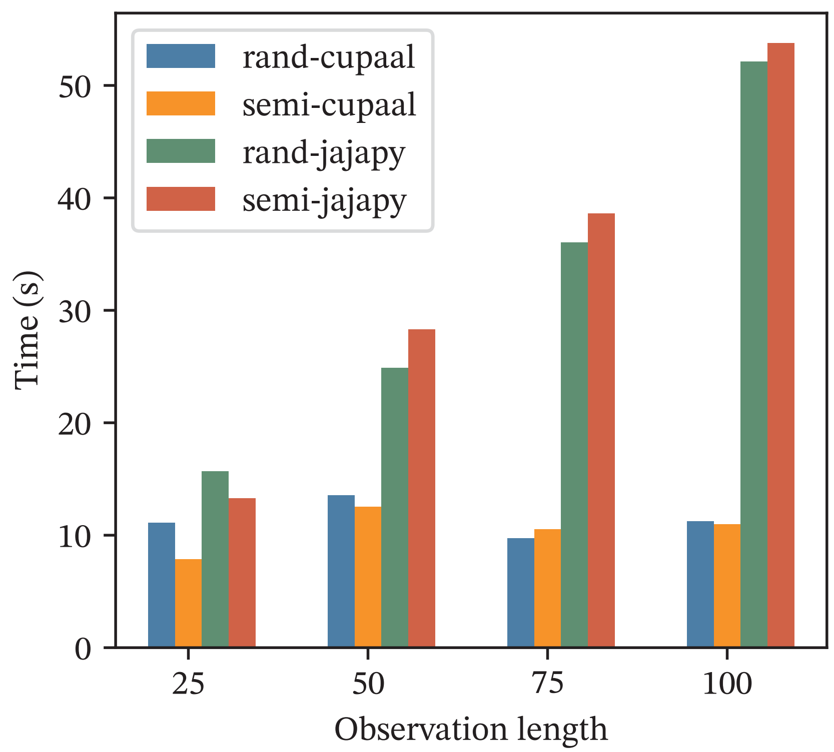


Tuning the initial hypothesis?

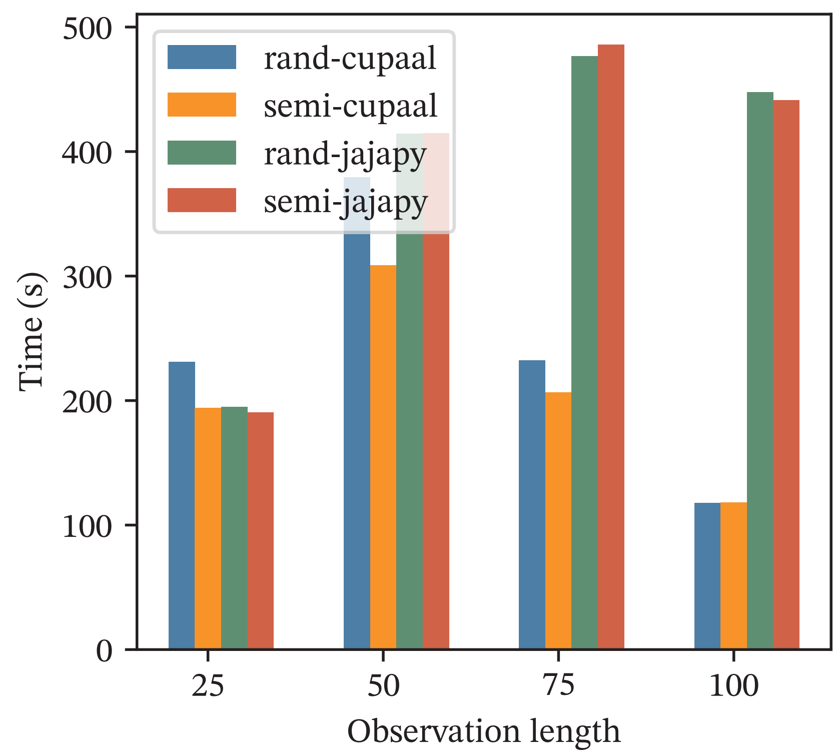
We played a bit limiting the image for the probability transition function of the initial hypothesis $\mathscr{M}_0$

We employed a semi-random generation of $\mathscr{M}_0$ which favoured structural similarities

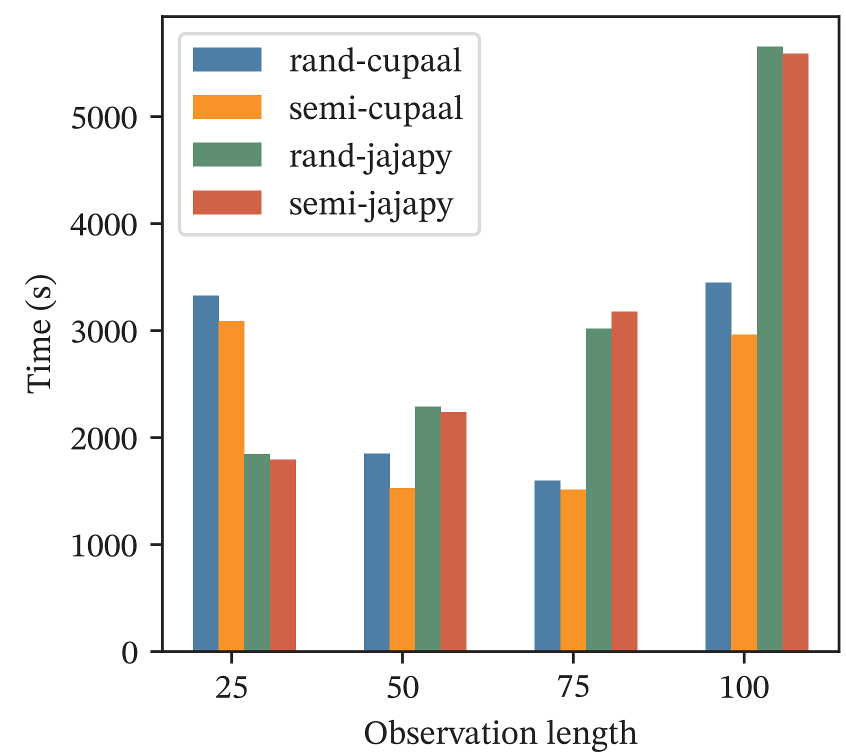
Remark: one needs to be careful here



Leader sync (N = 4 , K = 2)



Leader sync (N = 4 , K = 3)



Leader sync (N = 4 , K = 4)

Conclusion and Future direction

- Early to say that our approach using ADDs can outperform actual implementation of Jajapy (sparse matrices are quite effective).
- Deeper integration with Prism skipping intermediate model representations.
- Structural similarities might be better exploited on parametric models like those we used.
- It would be great trying out parameter estimation techniques on Stochastic Timed Automata

The End

The End

for today